APPENDIX A

```
/**<SCRIPT LANGUAGE="JavaScript1.2"> */

////////////////////////////////////////////////////////////////////
// Source Safe version and Date: DO NOT REMOVE
// $Header: /Versions/Current/Source/Collection/Client/ePulse/ePulse.js 45     4/10/00 10:34a
Rob $
//
// DO NOT REMOVE the LOGFILE and REVISION below. They are redundant
// with the $Header, but they are necessary because the Obfuscator will
// pass them through to the obfuscated script.
// $Logfile: /Versions/Current/Source/Collection/Client/ePulse/ePulse.js $
// $Revision: 45 $
////////////////////////////////////////////////////////////////////

var kl_isIE = navigator.appName.indexOf("Microsoft")!=-1;
var kl_isNav = navigator.appName.indexOf("Netscape")!=-1 &&
    parseInt(navigator.appVersion) >= 4;
if (kl_isIE || kl_isNav)
{
   /** BEGIN Customer Specific Information **/
   var kl_rootpath = "__rootpath__";
   var kl_companyid = "__companyid__";
   var kl_siteid    = "KL__siteid__";
   var kl_collectiongetURL = "__collectiongetURL__";


   /** These parameters are used in the following way:
   kl_rootpath is used to set the visibility of the session cookie.
      Its value is normally "/". But if there are two sites within the same
      domain, it may be different. For example, say there are two logical
      sites, one at http://ebusiness/Lucky and one at http://ebusiness/StaterBros,
      then to track sessions on these sites separately, the Lucky site would
      have to have its rootpath set to /Lucky and the StaterBros site would
      have to have its rootpath set to /StateBros.
   kl_siteid is the site Id in the database with the string KL prepended
   kl_collectiongetURL is the image file that the script does a "GET" on in
      order to send data to the server.
   **/
   /** END Customer Specific Information **/

   // this is a new one that should be added
   var kl_sessionCookieVisibility = ""; // this can be "" in most cases
   var kl_sessionCookiePathVisibility =  "; path=" + kl_rootpath;
   var kl_pageViewKey = "KL_PV"+kl_siteid;

   var kl_sessionCookieKey = kl_siteid;
   // Record in the  window the kl_startload time for the entire page, it
   // it isn't already set.
   // And record in this window the start load time of itself
   var kl_startload = new Date();
}


// Retrieve the named value from the cookie. If the
// value doesn't exist, return ""
function kl_getCookieValue(kl_valueName, kl_cookie)
{

   // s is start index
   var s = kl_cookie.indexOf(kl_valueName);
   if (s == -1)
      return "";
   s += kl_valueName.length;
   // e is end index
   var e = kl_cookie.indexOf(";", s);
```

```
    if (e == -1)
        e = kl_cookie.length;
    return kl_cookie.substring(s, e);
}

function kl_inDomain(refUrl)
{
    // from the URL: http://www.abc.com/..., extract
    // www.abc.com
    // Rules skip any non ":" characters, match starting with the
    // sequence ":\\", set the r[1] value to be the rest of the characters
    // up to the first "\"
    var r = refUrl.match(new RegExp("[^:]*:\\/\\/([^:\\/]*)")); /* */

    if (r != null)
        return document.domain == r[1];

    return false;

}

        // Install an event handler, If one already exists, replace it with
        // an anonymous function that calls both the original code and the
        // iEchoes hander.
        function kl_setHandle(kl_method, kl_newFunc, kl_oldFunc) // nf is newFunc, of is
oldFunc
        {
                //var kl_isNav = (navigator.appName.indexOf("Netscape") != -1);

            // look to see if there is any existing handler in place
                if (kl_oldFunc)
                {

                        var kl_ofuncBody = kl_oldFunc.toString();

                        // a stands for arg
                        // The regular expression pulls out the argument list of the old
                        // function, for example if the function string is "f(a, b, c) {var i
...}",
                        // then the match rule puls out "a, b, c" in a[1]
                        var a = kl_ofuncBody.match(new RegExp("\\(([^\\)]*)\\)"));

                        //if (kl_isNav && a != null)
                        //      kl_method = kl_method.replace(/\([^\)]*\)/g , a[0]);
                        //else
                        //      a = kl_method.match(/\(([^\)]*)\)/);

                    // if there were no arguments, manufacture as single argument e, which will
                    // become the event argument. The event argument is used by Keylime's code
                     // only when the browser is Netscape, but i doesn't hurt to create the
argument
                    // in all cases.
                        a = (a[1].length >0 ? a[1] : "e");

                        // replace the arguments in the new event hanlder to match
                        // the original event handler the user set in place.
                        if (kl_isNav)
                              // The reg expression below matches the argument list and the
replace
                            // replaces it with the new arg list.
                              kl_method = kl_method.replace(new RegExp("\\([^\\)]*\\)","g") ,
"("+a+")");

                        if (kl_ofuncBody.indexOf(kl_method) < 0)
                        {
                            // b is bracket character index
                             var b = kl_ofuncBody.indexOf("{");
```

```
                                if ((kl_isNav || kl_ofuncBody.indexOf("anonymous") > 0) && b >
0)
                                        kl_ofuncBody          =          kl_ofuncBody.substring(b,
kl_ofuncBody.length);
                                else if (kl_ofuncBody.indexOf("function") >= 0 && b > 0)
                                        kl_ofuncBody                                                =
kl_ofuncBody.substring(kl_ofuncBody.indexOf("function") + 9, b);

                                // escape any newline chars in the old function body, to exscaped
newlines
                                kl_ofuncBody = kl_ofuncBody.replace(new RegExp("\\n","g"), "\\n");
                                // replace any occurences of the this pointer to kl_this. The newly
defined
                                  // user function will be passed the original this pointer as
kl_this. This is
                                // necessary since the scope of the original code is being rplaced.
                                kl_ofuncBody = kl_ofuncBody.replace(
                                          new RegExp("([^a-zA-Z0-9$_])this([^a-zA-Z0-9$_])","g"),
"$1kll_this$2");
                                //kl_ofuncBody = kl_ofuncBody.replace(/\n/g, " ; ");
                                // Escape all ' characters
                                kl_ofuncBody = kl_ofuncBody.replace(new RegExp("'","g"), "\\'");
                                // escape all " characters
                                kl_ofuncBody = kl_ofuncBody.replace(new RegExp('"',"g"), '\\"');

                                // create then function body for the new event handler.
                                var kl_newFuncBody =
                                     "{" +
                                     // "window.alert('START ');" +
                                     "var a0 = '" + a + "';"  +
                                     "var ofb = '" + kl_ofuncBody + "' ;" +
                                     "var f = new Function( a0, 'kll_this', ofb);" +
                                     "var rv = f( " + a + ",this);" +
                                                    kl_method + ";" +
                                        //"window.alert('FINISHED !!!!!!!!!!!!!!l');" +
                                        "return rv;" +
                                        "}";


                                     return new Function(a, kl_newFuncBody);
                           }
                           else
                                     return kl_oldFunc;
                  }
                  return kl_newFunc;
         }


// force conversion to string
function kl_rts(URL)
{
   // URL MUST be either of type String or Location
    return URL.toString();

}


function kl_logerror(rs,hf,ln)
{
   // m stands for message throughout
   var m = "";

   if (rs.target != null)
   {
      m += "&rs=" + escape("Image load error.");
      m += "&hf=" + escape(kl_rts(rs.target.src.split("?")[0]));
   }
   else
```

```
    {
       m += "&rs="+escape(rs);
       if (hf != document.location)
          m += "&hf="+escape(kl_rts(hf));
       if (ln > 0)
          m +="&ln="+ln;
    }
    kl_log("ERROR", m);

    return true;
}


function kl_startmonitor(e)
{
    // w mean win
    var w = (kl_isIE ? window : e.target);

    // if the window has an instrumentation function, instrument it.
    // this will bypass calling instrument for frameset windows, but
    // all others will be instrumented.
    //if (win.kl_instrument != null)

    w.kl_instrument();

    // If there's data to send, send it. Normally there is data only if this is the first page
    // loaded in the session, and the SYSTEM record is what gets sent here.
    if (kl_amFirstWindowLoaded && !(kl_sessionrec  == null))
    {
       kl_submitlog("echo=" + kl_sessionrec);
       kl_sessionrec = null;
    }


    kl_log("LOAD", "");
}


function kl_addfocushandlers(o)
{
  o.onfocus = kl_setHandle("kl_startfocus(e)", kl_startfocus, o.onfocus);
  o.onblur = kl_setHandle("kl_endfocus(e)", kl_endfocus, o.onblur);
}

function kl_startfocus(e)
{
    var el = (kl_isIE ? window.event.srcElement : e.target);

    el.kl_focusstart = new Date();
    if (el.form != null)
    {
       if (el.name)
          el.form.myPeer.le = el.name;
       else if (el.id)
          el.form.myPeer.le = el.id;
    }

}

function kl_endfocus(e)
{
    var el = (kl_isIE ? window.event.srcElement : e.target);
    var kl_interval = 0;
    var now = new Date();

    if (el.kl_focusstart != null)
    {
```

```
         kl_interval += now.getTime() - el.kl_focusstart.getTime();
         el.form.myPeer.ft += kl_interval;
         if (el.kl_isSet == false)
         {
            el.form.myPeer.ns += 1;
            el.kl_isSet = true;
         }

         el.kl_myElementPeer.kl_value = kl_extractValue(el);


      }
      return true;
   }

   function kl_gotolink(e)
   {
      // l is link
      var l = (kl_isIE ? window.event.srcElement : e.target );
      var isad = false;

      if (kl_isIE)
      {
         var uc = l.tagName.toUpperCase();
         isad = (uc == "IMG" ? true : false );

         while (uc.indexOf("A"))
         {
            l = l.parentElement;
            uc = l.tagName.toUpperCase();
         }
      }

      var msg = "&lk="+escape(kl_rts(l));
      if (isad)
         msg += "&ad=YES";

      kl_log("GOTO", msg);

   }

   function kl_submitform(e)
   {
      // f is form
      var f = (kl_isIE ? window.event.srcElement : e.target);


      f.kl_lastelement = 0;
      f.kl_logformimpression("SUBMIT", f.myPeer);

   }


   function kl_resetform(e)
   {
      // f is form
      var f = (kl_isIE ? window.event.srcElement : e.target);

      f.kl_lastelement = 0;
      f.kl_logformimpression("RESET", f.myPeer);

   }

   function kl_logformimpression(reason, peer)
   {
      // NOTICE the with statement. The peer attributes used within are:
      // ne, name, ns, le ft
      with (peer)
      {
```

```
        if (ne > 0)
        {
           var m = "&rs=" + reason; // m is message

           // if the form has a name, report on it.

           m += "&nm=" + escape(name);

           // if the foucs time of the form was > 0, then report on it.
           if ( ft > 0 )
           {
              m += "&ne=" + ne;
              if (ns > 0)
                 m += "&ns=" + ns;
              if (le != -1)
                 m += "&le=" + le;

              m += "&ft=" + ft;

              // add the user-defd fields if there are any
              // the number of fields is limited to 10.
              var lim = 0;
              if (kl_InfoAry != null)
                 lim = kl_InfoAry.length;

              if (lim > 10)
                           lim = 10;
              for (var i = 0; i < lim; i++)
                 if (kl_InfoAry[i] != null)
                              m += "&i" + i.toString() + "=" + kl_InfoAry[i].toString();

              kl_log("FORM",m);
              // reset the peer fields
              ft = 0;
              ns = 0;
              le = -1;
           }
        }
     }
  }

  function kl_abortload(e)
  {
     var img = (kl_isIE ? evendow.event.srcElement : e.target);

     var m = "&ig=" + escape(img.src);

     kl_log("ABORT", m);

  }

  function kl_checkabort()
  {
     this.imgabort = 0;
     if (kl_isIE && document.images)
     {
        var ng = 5;
        for (var i = 0; i < document.images.length; i++)
        {
           var img = document.images[i];
           if (img.complete == false )
           {
              this.imgabort++;
              if (ng-- > 0)
                 kl_log("ABORT","&ig="+escape(img.src));
           }
        }
     }
  }
```

```
function kl_dounload(e)
{
   if (kl_formAry != null)
   {
      for (var d = 0; d < kl_formAry.length; d++)
         kl_logformimpression("ABANDON["+d+"]", kl_formAry[d]);
   }

   if (kl_checkabort != null)
      kl_checkabort();

   kl_log("UNLOAD", (imgabort > 0 ? "&ab="+imgabort : ""));
}

function kl_dobeforeunload(e)
{
   var win = (kl_isIE ? window : e.target);

   // if there's nothing to report just let it pass
   if (win.kl_logvalue == null)
      return;

   kl_dounload();

   if ( !(window.kl_logvalue == null))
      kl_submitlog("echo=" + window.kl_logvalue);

   if (kl_isIE)
      window.onbeforeunload = null;
}

function kl_timestamp(stamp, sys)
{

   var m = "&b=TSTAMP&st=" + escape(stamp.toGMTString())
      + "&zn=" + (stamp.getTimezoneOffset() * -1)            // need to reverse the sign
      + "&cd=" + kl_companyid + "&sd=" + kl_siteid + "&id=" + kl_theTopWin.klid + "&e=";

   var doc = document;

   if (sys && kl_inframe)
      doc = kl_theTopDoc;


   m+="&b=PAGE";
   if (doc.location)
      m +="&lc="+escape(kl_rts(doc.location));

   if (!sys && kl_inframe)
   {
      m +="&pt="+escape(kl_rts(kl_theTopWin.location));
   }
   if (doc.location.href != doc.location)
      m +="&hf="+escape(kl_rts(doc.location.href));
   if (doc.referrer)
      m += "&rf="+escape(kl_rts(doc.referrer));
   m += "&sq=" + escape(kl_theTopWin.klsq.toString());
   //m += "&sq=" + escape("1");


   m += "&e=";

   // dink the pageview sequence number in the cookie

   kl_theTopWin.klsq += 1;
   // rewrite the cookie
   kl_theTopWin.document.cookie =
```

```
                    kl_pageViewKey+"="        +        kl_theTopWin.klsq.toString()        +
kl_sessionCookiePathVisibility
                     + kl_sessionCookieVisibility;

   return m;
}

function kl_log(tp, s)
{
   var now = new Date();
   var delta = now.getTime() - kl_startload.getTime();

   var w = window ;
   if (w.kl_logvalue == null)
      w.kl_logvalue = kl_timestamp(kl_startload, tp == "SYSTEM");

   if (tp == "UNLOAD" &&
       (typeof(w.epulse_info) != "undefined")
       && (w.epulse_info != null))
   {
      w.kl_logvalue += "&b=" + tp + "&dt=" + delta + "&if="
                    + escape(w.epulse_info.toString())
                    + s + "&e=";
   }
   else
   {

      w.kl_logvalue += "&b=" + tp + "&dt=" + delta + s + "&e=";
   }
}

function kl_submitlog(tx)
{
   var d = document;

   if (d.ePingI == null || typeof(d.ePingI) == "undefined")
      d.ePingI = new Image();

   // setting the src property of the image causes the image to be loaded.
   // Load the image and deliver the goods.
   d.ePingI.src = kl_collectiongetURL+"?"+tx;

   return true;


}


function kl_instrument()
{
   var w = window;
   var d = w.document;

   // kl_submitlog function
   // The following code is netscape specific
   if (kl_useapplet)
   {
      // if using applet, set up applet as transmission method.
      //document.ePingA.registerTop(top);
      kl_submitlog = d.ePingA.submitlog;

      d.ePingA.setEPingPath(kl_collectiongetURL);
   }

   // set the kl_gotolink handler on all the links
   for (var i = 0; i < d.links.length; i++)
   {
      d.links[i].onclick = kl_setHandle("kl_gotolink(e)", kl_gotolink, d.links[i].onclick);
   }
```

```
    // set up to form data locations on the window. These locations will be used to store
    // collected information abotu the forms. This is necessary, only because under Netscape,
    // by the time the window receives an unload event the document is (often) destroyed.

    // hold the array of form peers
    w.kl_formAry = new Array();

    // holds slots 0 thru 9 of user-specified data (usually form data) for the page
    w.kl_InfoAry = new Array();

    // set up the forms handlers on all interior forms
    for (var j = 0; j < d.forms.length; j++)
    {
       var m = d.forms[j]; // just to save on characters

       // A form doesn't get focus!
       //kl_addfocushandlers(m);

       // set up the form infor array on the form

       // set up the window storeage area
       w.kl_formAry[j] = new Object();
       var n = w.kl_formAry[j]; // just to save on characters

       // make the form aware of the  storage area that maintains his storage
       m.myPeer = n;

       // ALERT, ALERT, ALERT
       // if a form has an element named "name" (a very stupid thing to do, but it
       // has been done) then the element overrides the
       // form.name property and makes the form name completely inaccessible to us.
       // Thus before recording the form name we have to discover
       // if its a string object and its not null

       if (m.name != null && typeof m.name == "string")
          n.name = m.name;
       else
          n.name = j;

       n.ne = m.elements.length;
       n.ns = 0;
       n.le = -1;
       n.ft = 0;
       n.elems = new Array();

       m.onsubmit = kl_setHandle("kl_submitform(e)", kl_submitform, m.onsubmit);
       m.onreset = kl_setHandle("kl_resetform(e)", kl_resetform, m.onreset);

       for (var e = 0; e < m.elements.length; e++)
       {

          kl_addfocushandlers(m.elements[e]);
          // set up the storage area
          n.elems[e] = new Object();
          // setup the element peer
          m.elements[e].kl_myElementPeer = n.elems[e];
          //n.elems[e].accumfocus = 0;
          n.elems[e].kl_focusstart = -1;
          n.elems[e].kl_isSet = false;
          n.elems[e].kl_value = kl_extractValue(m.elements[e]);
          n.elems[e].kl_name = m.elements[e].name;
       }
       m.kl_logformimpression = kl_logformimpression;
    }
}

function kl_extractValue(e)  // e is an input element on a form
{
```

```
    // extract the "current value" of the input element. Different "types" of input elements
    // are treated differently.

    // weed out bad params -- this code might not be necessary
    if (e == null || e.type == null)
       return null;

    var t = e.type.toLowerCase();
    if (t == "checkbox")
    {
        return e.checked;
    }
    else if (t == "text")
    {
        return e.value;
    }

    // recording values from other input types (such as SELECT are not supported at the moment)

    return null;
}


function kl_startsession()
{
    var w = window;
    var s = w.screen, n = navigator;
    var msg ="&sc="
        +s.width+"x"+s.height+"x"+s.colorDepth
        +"&bw="+escape(n.appName)+"&vs="+escape(n.appVersion)
        +"&pf="+escape(n.platform)+"&ua="+escape(n.userAgent);
    kl_log("SYSTEM", msg);
    w.kl_sessionrec = w.kl_logvalue;
    w.kl_logvalue = null;
}


/** START ePulse Initialization **/

// IMPORTANT the ';' below is NECESSARY for Netscape to be compatible with the
// obfuscated script
////////// DON NOT REMOVE//////////////////////////////////////////////////////
;
//////////////////////////////////////////////////////////////////////////////


if (kl_isIE || kl_isNav) // only allow Nets
{

    var kl_theTopWin = window;
    var kl_theTopDoc = document;


    // kl_inframe == true means we are not the top window
    var kl_inframe = (top != window);

    // define several property on this window. This is just to make sure
    // we don't have to deal with "undefined" value
    var kl_logvalue = null;
    var kl_sessionrec = null;
    var kl_useapplet = false;

    var kl_amFirstWindowLoaded = false;
    if (kl_isNav)
    {
       kl_useapplet = true;
       // use the base part of the collection URL as the code base
```

```
        var cb = kl_collectiongetURL.substr(0, kl_collectiongetURL.lastIndexOf("/"));

        document.writeln("<APPLET code=ePingA.class codeBase=\"" + cb +"\" id=ePingA"+
           " name=ePingA width=1 height=1 align=right MAYSCRIPT></APPLET>");
     }
     else
        kl_useapplet = false;

     // IMPORTANT: For netscape, this line HAS to come after the call to document.writeln(),
above.
     // It seems that document.writeln() causes netscape to set up some document properties that
     // otherwise wouldn't have been accessible until after the onload event occurs.
     // If you take the call to document.writeln(), above, out the following statment will cause
     // some kind of unhandable (in javascript) error such as access is denied.
     var kl_isSessionStarted = kl_getCookieValue(kl_siteid+"=", document.cookie);

     // kl_isSessionStarted is used to determine if this is the first (or one of the first in
framesets)
     // page view in the session. if there is not yet a session cookie, then this is one of the
first (
     // (or cookies might be off). If this is the first, then walk the hierarchy trying to get
at
     // the highest accessible (in same domain) parent window.

     //var ses = kl_getCookieValue(kl_siteid+"=", document.cookie);
     if ( kl_isSessionStarted == "" ) // only try to find the real top win if there is no
session cookie
     {
        // keep going only until we reach the top or our referrer is in a different domain.
           while(kl_theTopWin != top && kl_inDomain(kl_theTopDoc.referrer))
           {
              kl_theTopDoc = kl_theTopWin.parent.document;
              kl_theTopWin = kl_theTopWin.parent;
           }
     }

     // this processing relevant for fames. One window (the first window loaded)
     // is chosen to set up and determine the session housekeeping on the top window.
     //if (top.seenFirst == null)
     if ( typeof kl_theTopWin.seenFirst == "undefined" || kl_theTopWin.seenFirst == null)
     {
        kl_theTopWin.seenFirst = true;
        kl_amFirstWindowLoaded = true;
     }


     if (kl_amFirstWindowLoaded)
     {

        // The first window loaded sets up some things on the top window
        kl_theTopWin.kl_companyid = this.kl_companyid;
        kl_theTopWin.kl_siteid = this.kl_siteid;
        // set up the session id and sequence number storage areas.
        kl_theTopWin.klid = "";
        kl_theTopWin.klsq = "";
        if (kl_theTopWin.document.cookie != null)
        {
           kl_theTopWin.klid = kl_getCookieValue(kl_siteid+"=", kl_theTopWin.document.cookie);
           kl_theTopWin.klsq          =          parseInt(kl_getCookieValue(kl_pageViewKey+"=",
kl_theTopWin.document.cookie));
        }


        if (kl_theTopWin.klid == "")
        {
           //window.alert("Should be starting session");
           // take care of the session cookie
           kl_theTopWin.klsq = 0;
           kl_theTopWin.klid = "x" + kl_startload.getTime().toString();
```


```
        kl_theTopWin.klid   =   kl_theTopWin.klid   +   Math.random().toString().substr(2,20-
kl_theTopWin.klid.length);
        kl_theTopWin.document.cookie=                    kl_siteid+"="+kl_theTopWin.klid        +
kl_sessionCookiePathVisibility +
                                     kl_sessionCookieVisibility;
        kl_theTopWin.document.cookie= kl_pageViewKey+"=0" +kl_sessionCookiePathVisibility +
                                     kl_sessionCookieVisibility;
        kl_startsession();

      }

   }

   // the "this" qualification below is necessary. Some browsers
   // will require it, don't remove.
   this.onabort = kl_setHandle("kl_abortload(e)", kl_abortload, this.onabort);
   this.onerror = kl_setHandle("kl_logerror(rs,hf,ln)", kl_logerror, this.onerror);
   this.onload = kl_setHandle("kl_startmonitor(e)", kl_startmonitor, this.onload);
   if (kl_isNav)
      window.captureEvents(Event.ABORT | Event.ERROR );

   if (kl_isIE)
      // if not in a frame and IE, use the onbeforeunload handler
      onbeforeunload      =      kl_setHandle("kl_dobeforeunload(e)",      kl_dobeforeunload,
onbeforeunload);
   else // netscape doesn't have on beforeunload, so we are stuck with onunload (but the
applet
      // delivery mechanism solves the problems this presents.
     this.onunload = kl_setHandle("kl_dobeforeunload(e)", kl_dobeforeunload, this.onunload);

   // If the controller window (the amTopdog) is in a frame,
   // we have to adjust the event handling because the top dog might get
   // destroyed if his frame content changes.
}

/** END ePulse Initialization **/
////////////////////////////////////////////////////////////////////////
// WARNING: Do not change this copyright notice at all. It is a tag string
// that is looked for by the iEchoes Web server module Farm Monitoring process
/* Copyright 1999, 2000 Keylime Software, Inc. All Rights Reserved */
////////////////////////////////////////////////////////////////////////

/**<SCRIPT> */
```

EXHIBIT 2006